A methodology for the development of special-purpose function architectures

by RAYMOND A. LIUZZI*
Rome Air Development Center
Griffiss Air Force Base, New York

and

P. BRUCE BERRA
Syracuse University
Syracuse, New York

ABSTRACT

The research described in this paper concerns a generalized methodology for the development of special-purpose function architectures (SPFA). The development methodology can be used to introduce the concept of an SPFA approach to an organization.

The methodology provides an organized set of processes that can be followed to tailor the development of SPFAs to specific applications. This methodology consists of processes for identification, creation, testing, evaluation, and substitution of SPFAs. It permits a user to carefully select sets of database management functions as candidates to be moved from software into hardware, develop one or more SPFAs that perform this function, and evaluate the consequences of having the function performed as a new hardware architecture. A set of tools/components with which to carry out this methodology are included in the environment of a proposed database machine architecture development facility.

*This research was performed while the author was pursuing the Ph.D. degree at Syracuse University.
INTRODUCTION

Interest in computer architecture research, as applied to database management, has recently increased because of the advancing state of the art in inexpensive, fast new hardware components. Hardware technology is advancing primarily in three areas: central processing units (CPU), semiconductor random-access memory (RAM), and all-electronic bulk memories. The cost-to-performance ratio of CPUs will decline rapidly over the next 10 years. Low-cost CPUs with the performance capabilities of today's medium-priced minicomputers will be available for hundreds of dollars in the 1980s. New technologies in memories using bubbles or charge-coupled devices will rival existing fixed-head discs.

These trends have made it feasible to examine new hardware architectures that can perform database management system (DBMS) functions currently performed in software. How to determine which functions to implement in hardware and how to choose their optimal architecture, for a given user application, becomes a very difficult task. In order to help ease this transition, database machines have been introduced as new hardware architectures designed to perform DBMS functions.5

The notion of a database machine (DBM) has evolved primarily because of the need to accomplish database management tasks more efficiently. The evolution to the current class of DBMs can be traced by viewing Figure 1. DBMSs were originally developed to execute on large sequential systems and had to rely on the services of a generalized host execution system to perform many of their tasks. Examples of this class of system include the IDS system on an H6000, IMS on an IBM 360/370, and System 2000 on several large machines. 15 However, much of the processing efficiency of these systems is compromised by the inefficiency of I/O operations for processing data. The operating systems on these large machines must multitask a number of activities.

As minicomputer development progressed, it became apparent that many database management tasks could be accomplished more efficiently by removing them from the large sequential machine to a machine dedicated entirely to database management. Such a machine is called a back-end machine. Canady et al. outlined an architecture for performing various DBMS tasks on a back-end Digital Scientific Meta-4 Computer.7 Numerous advantages were cited, including security, reliability, and efficiency.

Further development in semiconductor technologies has produced the microprocessor and the microcomputer, along with the notion that it is economically feasible to develop computers that are primarily designed for database management. Previous research efforts have substantiated the fact that computer architectures that provide concurrency and content-addressing constructs can provide order-of-magnitude increases in the performance of certain database management functions.20,26 As a result, a new class of machines has emerged, with various unique architectures designed to provide these constructs. Liuazzi and Berra17 have defined a set of characteristics for a range of these DBMs that consists of the following:

1. An overall architecture composed of one or more special-purpose function architectures (SPFAs)
2. An architecture based on parallelism and content addressing
3. A set of compatible memory units for the storing and efficient retrieval of data
4. An architecture that is a back-end machine.

Several types of machines have been reported in the literature with these characteristics; some are given below.

The logic per track architecture of the University of Florida's Context Addressed Segment Sequential Memory Organization (CASSM) System,6 the multicell CASSM system,29 and the University of Toronto's Relational Associative Processor (RAP) system24,25 attain concurrency by moving logic from the central processing unit to the individual disk heads that read data from each track on a fixed-head disk. The RAP system has recently been extended to include a semiconductor charge-coupled device (CCD), random-access memory (RAM), or bubble memory.

The Ohio State Data Base Computer (DBC) proposed by Hsiao18 consists of a unique architecture that interconnects several specialized processors aimed at supporting secure large-scale databases. Each database is stored on content-addressable moving-head disk devices, and emerging technologies such as magnetic bubbles and CCDs have been chosen for part of the system.

The INFOPLEX system proposed by Madnick18 takes
advantage of new memory and processor technologies to organize a smart memory hierarchy to handle the storing and retrieval of information. Its information management functions are decomposed into a functional hierarchy implemented by a hierarchy of microprocessors.

The DIRECT system proposed by DeWitt is a multiprocessor organization for supporting relational DBMSs. DIRECT has a multiple-instruction, multiple-data stream architecture. It can simultaneously support both interquery and intraquery concurrency.

Associative processors have been experimentally examined for database management applications. Early studies by DeFiore, Stillman, and Berra using the Goodyear Associative Memory and later the STARAN Associative Array Processor established that searching a database was significantly improved by associative techniques.

The RELACS system proposed by Oliver is a DBMS using associative processors to implement the relational data model. RELACS is capable of supporting many functions of a database management system including retrieve, updating, deletion, modification, and addition.

In addition to this class of DBMSs, a set of specialized architectures have emerged. As Figure 1 indicates, these specialized architectures can form portions of a DBM. They are primarily designed to optimize a single database management function and are called special-purpose functional architectures. Several different types of database functions have been designed as SPFAs.

Roberts has proposed a specialized parallel computer architecture for high-speed searching of large textual files. The database to be searched is partitioned among independent high-speed serial-access memories that are searched in parallel by dedicated microprocessors connected to a common communication bus.

Hollaar has proposed a specialized merge processor that combines data from sorted input lists into a sorted output file. This processor is designed with architectural constructs that form the merge operation. Stellhorn proposed an inverted file processor that uses a specialized architecture to access files of document identifiers and performs the processing associated with a Boolean search request. Hollaar and Stellhorn propose a specialized architecture for textual information retrieval. The basic architecture of the system consists of several parallel search modules connected to a disk via a parallel/serial interface. This architecture is especially suited for list merging, updating, and sorting operations. Hollaar has also extended this work to include the design of a list-merging network.

Mukhopadhyay has proposed specialized hardware algorithms for nonnumeric computation. These algorithms can be implemented with various LSI technologies for high-speed pattern-matching needs.

Caprarol has proposed to integrate a data dictionary as an SPFA using associative processors. Singhania and Berra have designed a special-purpose function architecture using associative memories for pipelining a directory to a very large database. The results of this study indicated that the pipeline system provides faster retrieval than sequential inverted list systems, especially in the case of multiple-key retrievals. Karlowsky and Leilich from the Technical University of Braunschweig have proposed an SPFA called a search processor to search data stored on a mass memory without using the CPU and I/O of a host computer.

The Content Addressable File Store (CAFS) is a specialized hardware architecture that performs parallel processing techniques for implementing multifactor selection across either single files or the join of multiple files. This SPFA performs concurrent execution of powerful selection and retrieval functions on multiple data streams arising from the simultaneous reading of many disk channels.

The introduction of these SPFAs has provided users with a new approach to gaining specialized improvements in their database applications. Each of the SPFAs described can replace a DBMS function or functions currently being performed in software on a sequential machine. This notion that software functions can be either improved or replaced by hardware has been characterized as the SPFA approach. This paper examines the effect the SPFA approach can have on an organization, describes the need for an organized methodology to introduce the SPFA approach to an organization, and presents a generalized methodology that can be used to help in the development of SPFAs.

NEED FOR METHODOLOGY

The emergence of various types of SPFAs has prompted the need for an organized methodology that can be used to develop SPFAs for specific user applications. Typically, an examination of a DBMS shows that it is composed of several types of functions. First, a set of basic functions for each DBMS is used to manipulate data into a form acceptable to the application program. Examples of these functions are search, update, and modify. A second set of functions maintain data in a data dictionary or a database. The functions permit a logical expression of the database and maintain a physical access to the stored data. Next, a set of functions provide user interface capabilities via query generation modules and request generation modules. These functions provide various levels of natural user interface to a DBMS. Finally, a set of application modules is used to support functions that provide various editing capabilities to a DBMS user. Each of these functions are typically performed in software and are candidates to be developed as SPFAs.

If an organization wishes to seek ways to upgrade a current DBMS capability, it might want to introduce one or more of these functions as SPFAs in the form of hardware assist modules into its current environment. However, in order to exploit this SPFA approach fully, several questions must be examined:

1. What are the important factors to consider when choosing a function that can be developed as an SPFA?
2. What are the various algorithmic approaches and architecture considerations to implement the function in hardware?
3. How will new hardware technology affect the function's implementation in terms of performance, cost, reliability, and other relevant matters?
For each database management function that is a candidate for a move to hardware, several architectural options may be available. To evaluate each option, designers may have to build the actual hardware. If more than one architecture is being considered, several options may have to be built. Once these hardware options are built, procedures to test and evaluate them need to be established. However, if several SPFAs options exist for a given function, the actual hardware construction may not always be feasible because of the following three problems:

1. The expense of actually developing a number of hardware options
2. Time constraints
3. Inability to alter each SPFA easily after it has been built

Finally, several factors must be considered in choosing the actual hardware technology used to implement a SPFA. The technology chosen by a user depends on specific user application requirements. For instance, a comparison of competitive technologies that may be used for an implementation may indicate that one is faster than the other but is less reliable. Another factor may indicate that one may improve performance, but at a higher cost.

Thus, a generalized methodology can be very useful in providing an organized mechanism to introduce SPFAs for improving overall DBMS capability. The methodology must consider choice of DBMS functions, architecture options for the function, and implementation strategies for the function for each specific user application. A methodology has been developed and can be used in conjunction with a database machine architecture development (DMAD) facility. This methodology and a brief description of the DMAD facility are now described.

METHODOLOGY TO DEVELOP SPFAs

A methodology to develop SPFAs requires the following set of processes:

1. Select candidate function.
2. SPFA create.
3. SPFA test.
4. SPFA evaluate.
5. SPFA substitute.

The select-candidate-function process helps determine DBMS software functions that are candidates for replacement as hardware architecture SPFAs.

The create process transforms a description of each SPFA from a set of architectural considerations to a set of language statements. This set represents a functional description of an architecture that performs the DBMS function.

The test process functionally verifies that the SPFA performs the desired DBMS function. This process permits the DBM architect to examine the architectural constructs of the SPFA to insure that it meets its design goals.

The evaluate process enables the DBM architect to evaluate competing SPFAs. An architecture evaluation is conducted to generate performance timings of the SPFAs. These timings are based on using an assumed set of hardware characteristics to perform operations required by each SPFA.

Finally, the substitute process is composed of a set of procedures to enable the DBM architect to selectively integrate an SPFA within a DBMS capability. The substitute process helps the DBM architect assess the effect on the system of having a DBMS software function replaced by hardware.

The complete development is illustrated by a process flow model in Figure 2. The flow of this model indicates that the environment is initialized for each process request. This initialization configures the tools needed to complete a process. Several feedback loops are provided in this model to allow refinements during the development of the SPFA. These loops permit reuse of the complete set of tools for all processes. For instance, after an evaluation process is completed, the DBM architect may choose to alter an SPFA by modifying a portion of the architecture description. This may result in the recreation of the SPFA. Similarly, a single process can be repeated interactively so that an exhaustive series of tests or evaluations can be performed.

In addition, the final process, substitute, permits the DBM architect to assess the effect of the newly developed SPFA on the DBMS system. This process is used to integrate the SPFA into the DBMS and to help determine potential problems. The data collected following this process can help determine if the function is a logical candidate to be moved to hardware for a specific application.

The use of the process flow model also permits a user to tailor an SPFA development to a specific application. This helps insure that the developed SPFA meets the unique requirements of the application.

ARCHITECTURE/INTERFACES OF THE DMAD FACILITY

A database machine architecture development facility has been proposed as a specialized environment that hosts the tools and components needed to perform each of the processes in the generalized methodology. The DMAD facility consists of the following components, illustrated in Figure 3:

1. a service host machine (SHM) that is responsible for monitoring requests in the DMAD facility, staging input for the database function execution machine (DBFEM),
The SPFA is described in hardware description language. This MERGE function. For instance, illustrated in Figure 4 is a MERGE SPFA machine that is created from among several and reliability of the MERGE function for this application. are created as SPFA machines to perform the DBMS architecture options and is introduced to the DMAD facility. The current being performed in software, as part of a DBMS, on a sequential computer. However, several competitive new architectures can also perform this func­ tion. This organization needs to choose the merge hard­ ware architectures can also perform this func­ tion. An example of introducing the SPFA approach to an organi­ zation is shown in Figure 4. Assume that an organization... and providing a programming environment to described SPFAs. The SHM interfaces to network machines that may help in the execution of a SPFA. 2. a database function execution machine (DBFEM) that is responsible for hosting the execution of SPFAs as machines in the facility. This machine serves as a back end to the SHM and is capable of emulating a variety of computer architectures. 3. a master configuration control machine (CCM) that interfaces the SHM and DBFEM. This machine acts as the configuration manager for process requests to the DMAD facility. In this capacity the machine controls resources needed to support execution of an SPFA machine on the DBFEM. 4. a configuration identification machine (CIM) that interfaces to the SHM and is used to identify configuration requirements needed to execute SPFA machines. Specialized libraries are maintained and can be loaded on the CIM to help identify these requirements.

ILLUSTRATION OF THE SPFA APPROACH

An example of introducing the SPFA approach to an organization is shown in Figure 4. Assume that an organization requires improving performance and reliability in merging lists for its present applications. This MERGE function is currently being performed in software, as part of a DBMS, on a sequential computer. However, several competitive new merge hardware architectures can also perform this function. This organization needs to choose the merge hardware architecture that can optimize the overall performance and reliability of the MERGE function for this application.

Within a DMAD facility the merge hardware architectures are created as SPFA machines to perform the DBMS MERGE function. For instance, illustrated in Figure 4 is a MERGE SPFA machine that is created from among several architecture options and is introduced to the DMAD facility. The SPFA is described in hardware description language. This description is compiled and debugged to produce an executable version. When this version is complete, it is configured in the DMAD facility to produce a MERGE SPFA machine. During execution, complete control of the SPFA machine is maintained by a user with access to the facility. This permits the examination of all states of execution. Testing in this environment is done with a set of tools to verify that the SPFA performs its intended DBMS function.

After testing, an evaluation of the SPFA is performed. This evaluation consists of accumulating the time needed by the SPFA machine for a sequence of its operations. The timing of these operations is chosen by examining possible hardware implementations and associated timings.

For instance, in this example, since both performance and reliability improvements are sought, a user can examine hardware technologies that have high reliability characteristics as candidates for the MERGE SPFA machine’s operations.

In order to assess the effect of varying these choices, one or more realization assumptions can be described. Each choice becomes a separate realization of the SPFA and is used to generate separate sets of data on performance and reliability.

Once performance and reliability data are established for a specific SPFA description, the procedure described above can be repeated by varying some architectural features of the original SPFA description or developing one of the competitive MERGE SPFAs. This process can continue for a number of architectural options that may be available for this function.

Once a MERGE SPFA is chosen, the next development stage can be its substitution within a current DBMS. This process can be performed as illustrated in Figure 5. First, a computer system that can support a set of database management functions is referred to as a DBMS machine. Next, assume that this DBMS machine is emulated to execute in the DMAD facility. A DBMS software application program (SAP) is chosen to execute on the DBMS machine and call the services of the DBMS functions supported. The SAP typically calls a sequence of DBMS functions such as FIND, ORDER, MERGE, CLOSE, etc., as illustrated in Figure 5. Whenever the MERGE function is called, the option chosen for the

---

**Figure 3—Architectural/interfaces of the DMAD facility**

**Figure 4—Developing a special-purpose function architecture**
The MERGE SPFA machine is executed instead of the original MERGE function. The actual interfaces and the effect of substituting the MERGE SPFA can now be examined in terms of pertinent hardware/software tradeoff issues.

A positive assessment of the SPFA’s integration may lead the organization to choose to actually build a hardware prototype Merge SPFA.

**SPFA DEVELOPMENT METHODOLOGY PROCESS FUNCTIONS**

Described in this section are a set of procedures for the generalized methodology to help develop SPFAs. This methodology can be divided into the following phases:

1. identification of candidate DBMS functions
2. creation of SPFAs
3. execution of an SPFA machine

**Identification of Candidate DBMS Functions**

The identification of candidate DBMS functions to be moved from software to hardware is made by examining the typical requirements of a range of applications. Pertinent factors include current usage of the function, the ability to clearly define interfaces to the function, and the potential of the function for improving system performance and cost. This process requires examination of several functions that are portions of a current DBMS.

A DBMS machine identification procedure is used to configure emulations of machines that support current DBMSs. Each machine is configured with a DBMS, required system support software, and sample application programs. A set of DBMS functions within the DBMS are then identified as candidates for being replaced by hardware. These functions are currently performed by sets of software modules that are executed when the function is called.

A variety of criteria may be used in selecting candidate functions. These include the frequency of a function’s use (i.e., number of calls), the amount of time taken to execute the function, the complexity of the function in relation to other functions within the DBMS, and the potential for improvement in DBMS system quality if the function is moved from software to hardware.

Statistics on use of a specific function can be obtained by establishing break points at the entrance to the software module performing the function during execution. The rate of use can be determined from the number of times the break point is encountered. Each procedure is to use a performance monitoring tool to identify frequency of calls for a DBMS function. Such a capability may also be provided in conjunction with the description of each emulated DBMS machine by establishing a count for a specific instruction execution. For instance, the SMITE hardware description language provides a performance capability that permits a software monitor to accumulate the number of times an instruction is encountered during execution.

Once data on use are collected, the actual execution path of frequently called functions can be examined. This path is examined by actually stepping the execution of the function, instruction by instruction. This permits all entrances and exits to and from the function to be properly identified and documented. This procedure helps to identify the complexity of this function in relation to other functions and to identify all the interfaces required to and from the function within the DBMS.

Next, quality considerations may be examined. Such an analysis is based on assessing the overall improvement in the quality of the system that may be obtained by moving the function to hardware. For instance, will the movement of this function to hardware increase DBMS system performance but at the same time decrease the system's portability or reliability?

Questions such as these may be examined by establishing metrics for specific quality factors concerning the DBMS function. These metrics can be computed for such factors as reliability, maintainability, and flexibility. If the movement of this function to an SPFA can improve the quality of the DBMS in relation to a given application, then it may become a candidate function.

In summary, the choice of specific DBMS functions to be moved from software into hardware may be based on criteria such as use, performance and complexity, and quality improvements gained within the system. Once one or more candidates are selected, competitive SPFAs that can perform the desired DBMS functions must be examined.

**Creation of SPFAs**

A create process is selected to describe an SPFA that performs a candidate DBMS function. The objective of the create process is to translate a conceptual architectural description of an SPFA into an executable SPFA machine. This process consists of two procedures:

1. SPFA description development
2. SPFA introduction
The SPFA description development procedure requires the specification of a set of architectural constructs that, when executed as a machine, perform a DBMS function. These constructs can be specified in a hardware description language that defines a machine representation of an SPFA.

A specialized programming environment that exists within the DMAD facility is used to describe each SPFA in a hardware description language (HDL). This includes identifying the machine representation of the SPFA in terms of registers, interconnections, flow of information, and specific operations. Both control and concurrency dependencies among SPFA operations are described. As part of a description, each SPFA is created in such a way as to be fully compatible with the same interfaces as the software function it will replace. Once completed, a compilation procedure translates this SPFA description into a source and subsequently into an object file. The source file is debugged within the programming environment to eliminate source programming errors. If errors are identified, the SPFA description is modified and recompiled. This procedure continues until a correct SPFA is described. The SPFA compilation also produces an object file that consists of a set of microinstructions that are compatible with the DBFEM. These microinstructions are used to transform the facility into an SPFA machine using an SPFA introduction procedure.

This procedure identifies each SPFA to the DMAD facility. SPFAs introduced to the facility are entered into a database machine architecture configuration array (DMCA). This array is used to identify configuration requirements needed for executing each SPFA as a machine in the facility.

Execution of an SPFA Machine

An SPFA machine is ready for execution once all configuration resources are made available. These resources include access to the database function execution machine and any other specialized resource support. When an SPFA machine begins execution, the processes of testing, evaluation, and/or substitution can be performed.

The objective of the testing process is to determine whether the SPFA machine accurately performs the desired DBMS function. The testing process consists of execution of the SPFA machine by means of test cases, verification of the proper sequence of SPFA machine states, and debugging the SPFA machine.

In order to begin testing, a set of test cases are specified. They consist of specific input to the SPFA machine to ensure that it executes properly.

The SPFA machine executes by moving from state to state. A state can be specified at the SPFA source description level or at the microinstruction level. The microinstruction level permits identification of states at a much lower level than the source level. The level may be needed for detailed verification or debugging the SPFA machine. Types of testing capability include tuning, verification, probe, and visual examination of the machine.

One verification technique that can be used is the examination of the states of the machine at given instants of time. This examination can be conducted by establishing control points in the SPFA description. When these control points are reached during execution of the SPFA machine, a DBM architect performs an extensive verification of the state of the SPFA machine. For instance, registers, information resources, and control indicators can be examined. If they conform to preselected values, the state verification of the SPFA machine is established. However, if an error or inconsistency is found at one of these control points, the SPFA machine may not be verified, and debugging procedures are necessary.

A specific procedure that can be used to formally verify states of a SPFA has been proposed by Crocker. This procedure examines the execution of an SPFA machine and refers to a state change as a state delta. These state deltas are then examined in relation to predefined lists. If a state delta results in the formation of an improper list, the execution of the SPFA machine during the delta is questioned for possible error.

The debugging procedure performed in the facility for an SPFA machine consists of the identification of an error and the isolation of its causes by the DBM architect, who isolates the error by verifying the SPFA machine states until the error occurs. This isolation can be performed at the source description level or at the microinstruction level, if necessary, to insure that the error is found.

During debugging the DBM architect views the actual representations of the SPFA machine. This permits all name variables and conditions in the SPFA description to be inspected. During the debugging exercise the DBM architect has complete control of the SPFA machine and can step it through various execution states.

After the SPFA machine has been tested, the evaluation process may be requested. The objective of this process is to evaluate SPFA machines by using hardware operations identification and SPFA performance procedures.

The hardware operations identification procedure examines each SPFA description to identify specific hardware operations. A realization assumption (RA) defines a set of specialized hardware implementation characteristics for these operations, and these characteristics are used to generate timings for the specific hardware operations identified for an SPFA. The times are entered into a realization timing control array (RTCA) for each realization assumption. Each SPFA machine is then executed, and the results of separate executions are collected and analyzed.

Once an SPFA has been evaluated, the final execution process is substitution. The objective of this process is to investigate the effects of substituting an SPFA for a candidate software DBMS function. The substitution process consists of DBMS/SPFA machine identification, selective integration, and evaluation.

The DBMS/SPFA machine identification procedure consists of identifying the configuration requirements used to transform the DMAD facility into a DBMS/SPFA machine. A set of requests are executed on the DBMS/SPFA machine that call the supported DBMS functions. These functions are performed by the appropriate sets of software modules. However, when a request is made for the specific DBMS function supported by the SPFA, the SPFA machine is automatically called to perform the DBMS function.

A DBMS/SPFA machine selective integration function pro-
procedure performs the actual substitution of the SPFA machine. A virtual database machine monitor (VDMMM) can be used as the tool for this procedure. The VDMMM is designed as a virtual machine monitor that supports control of both a DBMS machine and an SPFA machine. Each time an SPFA machine is called, it completes the DBMS function, then returns control to the DBMS machine.

A DBMS/SPFA machine evaluation procedure enables a user to assess the impact of integrating an SPFA machine and a DBMS machine. This procedure includes executing the same set of requests to supported DBMS functions with and without the presence of the SPFA machine. This procedure is used to identify any further interface problems that may result from the presence of the SPFA machine, helps assess the system feasibility of having the function performed as an SPFA, and enables performance comparisons to be made.

CONCLUSIONS AND FUTURE RESEARCH

The effect of continuing advancements in hardware technology is promoting the feasibility of having SPFAs perform many database management functions.

This notion, referred to as the SPFA approach, can serve as a vehicle for increasing the overall DBMS capability in an organization. A DMAD facility, used in conjunction with the methodology defined in this paper, can serve as a specialized model to help introduce SPFAs to an organization via the SPFA approach. This methodology is organized and presented as a set of specific processes. Each of these processes is designed to permit a user to tailor the development of an SPFA to a specific application.

A set of tools/components to perform specific procedures for this methodology are also included in the proposed environment of the DMAD facility.

The extensive use of this methodology can also be directed toward examining critical tradeoff issues for defining a proper hardware/software mix in an overall system for a specific application. The methodology, used in this fashion, can serve as a vehicle to help choose which functions should migrate from software to hardware from an overall system architecture view.

In order to use the methodology in system optimization, further research is needed to expand use of specific optimization methods that can be used to formulate a direct relationship between SPFAs and sets of user requirements. In order to assume this role, modeling techniques may be added as procedures to follow each of the development processes of the generalized methodology. These procedures include several evaluation techniques, among them mathematical modeling and simulation. Some specific techniques that can be used in a hardware/software system tradeoff have been proposed by Vemuri. Further work is needed to expand the detailed use of these techniques.

REFERENCES


